04

Tratando el error 404 #2

Transcripción

[00:00] ¿Qué más? Necesito también anotarlo con restControllerAdvice y cuando ya comiencen a ver un poco más de Spring Open, todo lo que es de programación orientada a aspectos, van a entender mejor lo que es un Advice.

[00:22] Por ahora, lo que les voy a decir es que esto actúa como una especie de proxy para todos nuestros controllers, por algo está como rest controller, para interceptar las llamadas en caso suceda algún tipo de excepción. Ya si ustedes ya conocen o exploran un poco de programación orientada a aspectos, van a entenderlo mucho mejor. Por ahora lo vamos a dejar así porque no es el propósito del curso.

[00:43] Y bueno, por el momento esto es suficiente como para interceptar nuestros métodos de controller y atrapar alguna excepción que sea lanzada. Ahora, si yo quiero devolver un código de error en específico y yo ya sé que ResponseEntity es la entidad que tienen los tipos de error mapeados dentro y qué hace un wrapper de esta respuesta y la retorna cliente, ¿qué es lo que necesito aquí? Exacto.

[01:18] Un public ResponseEntity. No voy a poner ningún tipo de dato en el generic, porque este sí es genérico, y especies genérico no necesita especificar un tipo de objeto especial y le voy a poner tratarError404. Y lo que le voy a decir aquí es return ResponseEntity.notFound. Y eso debería ser todo.

[01:48] Bueno, me está dando un error y es porque falta el .build al final. Ahora al igual que los métodos en el controller, que lo vemos aquí, yo necesito

avisarle a Spring, que este método tiene que ser llamado en caso alguna excepción de algún tipo sea detectada, para eso es una anotación llamada @ExceptionHandler.

[02:20] Y en @ExceptionHandler lo que yo le voy a decir es el tipo de excepción que yo quiero tratar. En este caso es esta de aquí: EntityNotFoundException, un espacio para importarlo .class. Entonces con eso le digo "ExceptionHandler, cuando tú en este RestControllerAdvice identifiques que EntityNotFoundException es lanzado, como estás anotado como @ExceptionHandler, entonces vas a retornar este código de aquí, esta respuesta de aquí, con notFound.

[02:58] Vamos a esperar que recargue nuestro servidor, parece que ya recargó. Ahora sí cargó, estoy seguro, y regreso aquí. Vieron que estaba con 500 internal server error, vamos a ver qué sucede ahora y en efecto ahora tenemos 404 not found y obviamente ningún tipo de contenido porque no encontró nada. Pero de esta forma ya pueden ver cómo usando el @ExceptionHandler, con un controllerAdvice, ya podemos ya personalizar mucho más el comportamiento de nuestra aplicación.

[03:35] Ahora ya sabemos que este error no es un error de servidor. Es un error que es simplemente porque tu registro no existe y es por una excepción en específico. ¿Qué otro tipo de errores ustedes creen que podemos encontrar aquí en nuestra aplicación?

[03:47] Yo les doy uno. Cuando hacemos la validación, estamos muy propensos a tener errores porque los campos que ingresamos o no son válidos o los que son requeridos no están presentes. Y eso nos retorna muchas veces un bad request. Vamos aprender cómo tratar ese tipo de errores en el siguiente video.